Porting MPlayer to MINIX 3

Pieter Hijma - hphijma@few.vu.nl

November 23, 2007

Contents

1 Introduction

This document describes the MPlayer to MINIX 3.1.3 port. The goal is to create a basic MPlayer port for MINIX.

The order in which everything is described in this document is not the order followed during the period of porting. First a feasibility study was done after which the port could start. MINIX on VMWare was chosen as platform because of the emulation of the `es1371` soundcard. After configuring, compiling and linking, video was working. Then a new module was written for MPlayer to connect MPlayer to the audio framework of MINIX. Because the audio/video synchronization was very bad which might be caused by the performance of VMWare, a search for hardware began. It was rather difficult to find a `es1371` card and a videocard that works with the MINIX 3 port of X11. After these cards were found, the audio/video synchronization problems were solved and a new driver was written. Because of some problems with X11 on MINIX 3.1.2a, the audio drivers were ported to MINIX 3.1.3. Because GCC has support for 64-bit integers, the include files were changed only for GCC and not for ACK. However, this is an unwanted situation and to make sure that the include files were exactly the same, some workarounds were made. Finally a package was made in the MINIX easypack format.

The following sections describe the feasibility study, installation of MINIX 3.1.3 on various platforms, the prerequisites for MPlayer to run and the actual port. The last sections are written in a tutorial kind of way, so that someone could perform the same steps to improve the port for example. After this, some problems are described and some future improvements are discussed.

2 Feasibility

Before to start the project it is necessary to look into what MPlayer needs from the operating system. This is done by looking into the code of MPlayer and trying to find all system calls and `ioctl()` calls that are necessary to let MPlayer run.

It is soon clear that the three possible problems consist of timing, video out and audio out.

2.1 Timing

The timer is used to play the video and audio at the right speed and to keep the audio and video in sync. MPlayer offers several ways to do this. One of them uses the `gettimeofday()` function. This will certainly work on MINIX 3.

2.2 Video out

Video out looks like the most difficult problem, but MPlayer offers again a whole set of options. There is a possibility to use the functions of X11 and there is a also something as vidix, which is a video interface for unix. X11 will probably provide enough functionality for MPlayer, although MPlayer is optimized for XFree86, whereas MINIX has a port of Xorg.

2.3 Audio out

This is what is going to give the most problems. MPlayer can handle different kinds of audio libraries like ALSA and OSS, and Sun audio. MINIX 3 has a different of framework, but it looks like the OSS and Sun frameworks. There are several solutions.

It is possible to extend the audio framework in MINIX 3 in such a way that it supports ALSA or OSS. Another solution is to write a module for MPlayer that implements the MPlayer interface and calls the right MINIX 3 functions. There is a problem with this. MPlayer likes to know how much it can write into the audio buffer before blocking, but there MINIX 3 has no such thing in its interface to know this. This can be solved in two ways.

The first solution is implementing a `select()` call that tells if it is possible to write into the buffer without blocking. The problem with this solution is that `select()` doesn't say how much data can be written. Another solution is to implement a `ioctl()` to the audio framework that says how much is in the buffer.

2.4 Conclusion

The conclusion is that MPlayer looks very portable. The audio will probably be the biggest problem, but it seems solvable. Overlooked during the feasability was the problem with 64-bit integers. MPlayer needs support for this, but in MINIX there is little support for 64-bit integers. GCC offers the `long long` types with which workarounds can be made.

3 Installing MINIX 3

Boot the PC with a MINIX 3 cd-rom and press ESC to get into the boot-loader. The used PC has a problem with DMA and the harddisk, so type `ata_no_dma = 1` and then `boot` to startup MINIX. Log in as root and type `setup` to start the installation of MINIX 3 on harddisk. Choose the right ethernet card and type `expert` to control setting up the partitions. Choose the right disk with `+` and try to read it with `r`. In this case the right disk is `/dev/c0d4`. Go to an empty partition and make the type `81` and choose an appropriate size, in this example `3000000 Kb`. A right value for the cylinder has to be chosen to make it work.

Then press `w` to save the partition table and `q` to exit. Type the name of the primary partition, which is in this case `/dev/c0d4p3`. To be on the save side, the size of the home directory is made `1500 Mb`. (The `/usr` and `/home` filesystems will have the same size.) Let the system scan for bad blocks and let it install the files.

After copying you can type `shutdown` to get into the boot monitor. Press Ctrl-Alt-Delete to reboot the system. After rebooting, the system boots into the boot monitor on partition `c0d4p0` which is called `d0p0s0` by the boot monitor (`s0` is a subpartition). As this system has more MINIX 3 installs, an entry in the boot menu has to be created. Type `set` to see all the environment variables. Create a new entry by typing `new(4,Start MINIX 3 final) {boot d0p3s0;}`. Save the environment by typing `save`. This command makes sure you get into the boot monitor of the new MINIX 3 install. Type `exit`, which starts this boot monitor and then `4` to get into the boot monitor of the new MINIX 3 install.

In the boot monitor of the new MINIX 3 install press `ESC` and type the command `ata_no_dma = 1` and `save` to save this environment variable that makes sure the harddisk driver doesn't use DMA. After typing `exit` and choosing the right installs, MINIX 3 boots.

Login as root and create a password by typing `passwd`. Create a normal user by typing `adduser pieter other /home/pieter` and `passwd pieter`.

4 Prerequisites

4.1 Software packages

The software packages needed to compile MPlayer are `glib-1.2.10`, the package `binutils-2.16.1`, `make-3.80`, `gcc-4.1.1` and `X11R6.8.2`. You can install these by using the `packman` command as root user. There might be some dependency issues, so this order is recommended.

4.2 Library update

MPlayer needs the `double rint(double x)` function. The file `rint.c` is added to `/usr/src/lib/math` and the `Makedepend-ack`, `Makedepend-gnu` and the `Makefile` are updated. The `math.h` in `/usr/src/include` is updated as well.

Compilation of the libraries is done by doing:

```
cd /usr/src/lib/math

make clean

cd /usr/src/lib/

make depend
make all
make install
```

First add `/usr/gnu/bin` to the path in `/root/.ashrc`. Then do:

```
make depend-gnu
make all-gnu
make install-gnu
```

MPlayer needs a math constant that is not defined in `mathconst.h`. This file is updated as well.

4.3 Audio drivers

The package `audio-1.0.0` has two drivers. The driver for the SoundBlaster 16 card is for an ISA card. Therefore, it is not very usefull for MPlayer. The `es1371` driver is for a PCI card, which is emulated by VMWare. The driver is based on the OSS (Open Sound System) driver, but it implements the MINIX audio framework. Unfortunately, there is no volume control in the driver.

Because qemu emulates the es1370 driver and this driver is rather similar to the es1371 driver, a new driver is created from the es1371 driver. Both drivers

are updated to work with the safecopy construct in MINIX 3.1.3 and volume control is added.

To install the audio drivers, copy the directory `audio` to `/usr/src/drivers` and the file `ioc_sound.h` to `/usr/include/sys`. Go to the `es1370` or `es1371` directory in `/usr/src/drivers/audio` and execute the following (see the `README` file):

```
make depend
make
make install

cd /dev
mknod audio c 13 0
mknod rec c 13 1
mknod mixer c 13 2
chmod 666 audio driver mixer
```

Finally copy the file `drivers.conf` to `/etc` (or make sure the entries for `es1370` and `es1371` are available) and recompile and reinstall everything in `/usr/src/ibm`.

4.4 X11

4.4.1 On real hardware

The video card used is a PCI Matrox Graphics, Inc. MGA 2064W [Millenium] rev 1 with `0x102b` as vendor id and `0x0519` as device id.

Because of the fact that the machine used only has two PCI slots and the video card is a PCI video card, the ethernet card has to be removed to make place for the video card. X11 needs a network to run, so a loopback device has to be setup. This is done by editing `/etc/inet.conf`. The contents should be `psip0 {default };`.

X11 automatically detects the video card and uses a build-in configuration file which uses the VESA driver.

X11 is started as normal user with the command `startx` and with the `.xinitrc` file with contents `twm`.

5 Porting MPlayer to Minix

5.1 Configuring

To configure MPlayer it is necessary that the normal user has `gcc` in his path. Add `/usr/gnu/bin` to the path in `.ashrc`. The `configure` script is a `sh` script that is not able to finish with the normal `/bin/sh`. Instead `/bin/bigsh` is used that has more stack memory.

Configuring MPlayer is done with the command:

```
bigsh configure --charset=US-ASCII --disable-network \
  --with-x11libdir=/usr/X11R6-gcc/lib --disable-vidix-internal \
  --enable-runtime-cpudetection --disable-mencoder
```

The option `--charset=US-ASCII` has to be added because no working `iconv` program is found. The option `--disable-network` is used because the define MSG_OOB is not known in the file `stream_ftp.c` and because stream caching is disabled. The option `--disable-vidix-internal` is used because `libdha` cannot compile without `sys/mman.h` and finally `--disable-mencoder` is used because the `flockfile()` functions are not available in `stdio.h`.

The script uses the small program `cpuinfo` in `TOOLS` to detect the cpu. This program cannot compile because of some 64-bit integer problems. Adding the following on line 853 in the configure script helps:

```
-D'i64_t=long long'
-D'int64_t=long long'
-D'uint64_t=unsigned long long'
```

Because of the fact that the program `tr` has different interfaces in MINIX 3 and other operating systems, the configure script doesn't execute correctly. It does not include all the codecs. The following has to be adjusted:

```
tr '[a-z] ' '[A-Z]\n'
```

The last expression has to change to `'[A-Z]\012'` to make it work (octal value for `'\n'`.

It might be the case that the file `crtso.o` is not found by `/usr/gnu/bin/gld`. Then the libraries have to be recompiled in the directory `/usr/src/lib` for gcc (`make all-gnu`, `make install-gnu`. There are also a lot of errors for `ldd`. This is not a problem. It doesn't affect the configure script and there are no shared libraries on MINIX 3.

To connect MPlayer to the audio framework provided by MINIX 3, a new module `ao_minix.c` is created that implements the interface that is requested by MPlayer. Unfortunately, the MINIX 3 framework doesn't provide enough functionality to satisfy MPlayer, so the framework is extended with the following `ioctl`'s: `DSPFREEBUF` is added because MPlayer wants to know if there are free buffers available. Because the video is synchronized to the audio, it also wants to know how many samples are currently in the buffer. This can be requested with `DSPSAMPLESINBUF`. Finally two primitives to pause and resume the playing of sound are added (`DSPIOPAUSE` and `DSPIORESUME`). Some changes are also made to make sure MPlayer can do seeks in audio streams.

The configure script is updated in such a way that it recognizes if the new audio framework is supported. It will then include the `ao_minix.c` module. The file `libao2/audio_out.c` is changed to include the `ao_minix` functions.

There are some error messages defined especially for the MINIX audio module. These are defined in the file `help/help_mp-en.h`.

Other changes to the configure script are the change from `ar` to `gar` to create archives, the undefine of `USE_STREAM_CACHE` which is necessary for the compilation process and the use of a special macrofile to set global defines.

Finally the file `version.sh` is edited to make sure that the `VERSION` is defined in such a way that it includes `MINIX 3.1.3`.

5.2 Compiling

To compile MPlayer the command `gmake` has to be executed.

The first problem that occurs has to do with the 64-bit integers. MINIX 3 has only the `u64_t` defined in the include files, but `gcc` has internal support for 64-bit integers. This problem is solved by using a global macro file that modifies every call to 64-bit integer types to the basic 64-bit types `long long` and `unsigned long long`. The statement `-Iminix.macros` is added to the `CFLAGS`.

The helper program `codec-cfg` also needs the 64-bit integers. The `Makefile` is edited to include the 64-bit defines.

The MPlayer source contains a lot of inline assembly statements. Some compilers need extra `'_'` just before the variables. This is handled in the file `mangle.h`. MINIX 3 has to be added to the list.

The problem that MINIX 3 does not have the `mmap()` functions (in the file `sys/mman.h`) can be solved by undefining `USE_STREAM_CACHE` in the configure script which puts the statement in `config.h`. This gives some strange compilation errors in the file `stream/cache2.c`. The function `stream_enable_cache()` is changed to `1()` by the preprocessor. This is solved by putting `#ifdef USE_STREAM_CACHE` around the function.

In the file `stream/stream_ftp.c` `fd_set` is undeclared. This can be solved by adding an `#include <sys/select.h>` statement to the file. The constant `MSG_OOB` is also not known. This is a TCP/IP thing. Because there is no stream cache and because of these errors, the `--disable-network` configure option is used.

The file `osdep/getch2.c` needs an `#include <sys/select.h>` statement to compile.

The files `libvo/x11_common.c` and `libvo/vosub_vidix.c` contain `#include <sys/mman.h>` statements. These can be removed without problem.

The file `libfaad2/Makefile` needs to include the underlying directory as well to compile this library. `-I..` is added to the `CFLAGS`.

The library `libdha` uses the `mmap()` functions. Compilation of this library is disabled by adding `--disable-vidix-internal` to the configure script.

5.3 Linking

In the file `stream/stream.a` made of `stream/cache2.o` and `stream/stream.o` the functions `stream_fil_buffer()` and `stream_seek_long()` have multiple definitions. This is solved by putting `#ifdef USE_STREAM_CACHE` statements in `stream/cache2.c`.around the functions `cache_stream_fill_buffer()` and `cache_stream_seek_long()`

In `libswscale/libswscale.a` made of `libswscale/swscale.o` are undefined references to variables that are used in inline assembly statements. It turns out that `MANGLE` is redefined in `libavutil/internal.h`. This should be changed to include MINIX as well.

The same error occurs in the `libpostproc` and the `libavcodec` directory. These libraries just have to be recompiled with the new `libavutil/internal.h` file.

5.4 Packaging

A package is made by making a `build.minix` file. The contents is:

```
#!/bin/sh
PATH=/bin:/usr/bin:/usr/local/bin:/usr/gnu/bin

/bin/bigsh configure --with-x11libdir=/usr/X11R6-gcc/lib \
    --disable-network --disable-vidix-internal \
    --enable-runtime-cpudetection --disable-mencoder \
    --charset=US-ASCII
gmake clean
gmake
gmake install
```

There is a `.binpackage` file with contents: `binsizes=xxl` and a `.descr` file with contents `MPlayer - The Movie Player`.

6 Problems

MPlayer has bad performance when the system is loaded. According to Jorrit Herder this is a scheduling problem. It would be interesting to see what happens on a faster video card.

MINIX 3 doesn't save the contents of floating point registers on a context switch. The compilers don't generate code that uses the floating point registers, but because MPlayer contains a lot of assebly code, it might be that some part of MPlayer uses the floating point registers. If this is the case and MPlayer is used simultanuously with an application that also uses the floating point registers, this will lead to problems. It seems that running two instances of MPlayer doesn't give a problem.

The man page of MPlayer doesn't work properly with the nroff command in MINIX 3.

7 Future improvements

When `mman.h` becomes available, MPlayer can be compiled with stream caching and vidix support. Network support can then be enabled if the constant `MSG_OOB` is known, which is for processing out of band data.

Probably OSD (On Screen Display) support can be compiled when the FreeType libraries are available.

Compilation of `mencoder` succeeds, but in the linking phase no references to `flockfile()` `funlockfile()` can be found. These belong to `stdio.h`.

An improvement for the audio would be if a OSS port would be available or that MINIX implements the interface of OSS or ALSA.